Current Status of Development of New VLBI Data Analysis Software

S. Bolotin, J. Gipson, D. Gordon and D. MacMillan

Abstract At the 2010 IVS GM in Hobart we proposed a design for our next generation VLBI data analysis software. In this paper we review the current status of this software and plans for future development. We also demonstrate the current capabilities of the software.

Keywords VLBI, data analysis, software

1 Introduction

Due to the increased number of VLBI observations and participated stations, implementation of new VLBI2010 technology will require a sophisticated approach to data analysis. The current CALC/SOLVE system for VLBI data preparation and analysis should be replaced with more flexible, power and user friendly software.

The first step in this direction was made in 2007 when the IVS Working Group on VLBI data structures (IVS WG4) was established. The results of the work of the IVS WG4 made it possible to start work on developing the new VLBI data analysis software. In August of 2009 the VLBI group at the NASA GSFC started this development.

The core requirements of the software and its general design and architecture overview were presented at the IVS General Meeting at Hobart (Tasmania) in February, 2010 (Bolotin et al., 2010). After discussion of the software design at the meeting, the whole architecture was reviewed and slight modifications were made. In March, 2010 the first lines of the source code were written.

The first executable that will replace a part of the current CALC/SOLVE system will be νSolve, an interactive part of the SOLVE system that is currently used in preliminary data analysis.

In this article we will cover the current status of the software development process. Section 2 describes tools that are used in the development process. In Section 3 we provide an overview of the modular design of the software. The current functionality is described in Section 4. In Section 5 we outline our plans for the future.

2 Software development environment

The software is written in the *C++* programming language. It uses the *Qt* library for high level data abstraction, the system's *libc* library for low level system functions and the *libm* library for standard mathematical procedures.

The new VLBI data analysis software will be distributed as source codes. To make the software distribution portable we use the *autoconf*, *automake* and *libtool* packages. These packages are parts of *GNU Build System*. With this we can create a highly portable software distribution, adjust the distribution to local needs, and minimize end-user effort to compile the package.

As a text editor we chose the *Geany* software. It has the following advantages. *Geany* possesses basic integrated development environment (IDE) features: syntax highlighting, code folding, symbol name autocompletion, code navigation and build system. It de-

Sergei Bolotin, John Gipson, David Gordon and Daniel MacMillan
NVI, Inc./ NASA Goddard Space Flight Center, 8800 Greenbelt Road, Greenbelt, Maryland 20771, USA

pends on only a few external packages and is independent of distributives. It is also small, lightweight and fast. *Geany* is known to run under Linux, FreeBSD, NetBSD, OpenBSD, MacOS X, AIX v5.3, Solaris Express and Windows operating systems.

We are also using *Doxygen* software to generate a reference documentation from source tree. *Doxygen* is a documentation system for *C*, *C++* and many other programming languages. It can automatically generate reference documentation in different formats: HTML, man pages, LaTex, RTF, PDF, etc. The documentation is generated directly from the source code, which makes it consistent with the current source tree.

Currently, the VLBI data analysis software consists of two parts:

1. *Space Geodesy Library*, a library where data structures and algorithms are implemented (about 90% of total source code);
2. an executable *vSolve* – a driver that calls the library and organizes work with an end-user (about 10% of total source code).

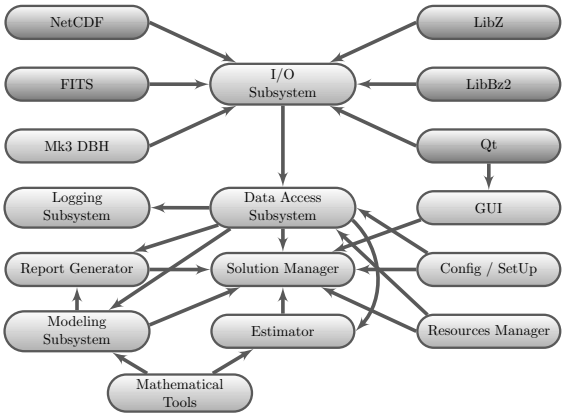

Fig. 1 Modular design of the new VLBI data analysis software

3 Modules

To be stable and flexible, our system has a modular design. A module is a logical block of code that is loosely tied with other parts of the software.

The general modular structure of the software is presented on Fig. 1. Each arrow on the figure represents a dependency that provides information (data types, function calls, constants). Only the main dependencies are shown on the figure.

Several modules in the figure (*FITS*, *NetCDF*, *LibZ*, *LibBz2* and *Qt*) represent external libraries.

The sizes of the modules could vary. There are small blocks, like *Logging system* with a few hundred lines of source code and commentaries, and large modules, such as *Modeling Subsystem* with tens of thousands of source code lines. Not all modules currently are completed, and some modules will be realized later. Now, we will give an an overview of currently implemented modules.

Logging subsystem is a small module that implements the system logging functionality. The *Logging subsystem* is capable of receiving a message from other parts of the software, filter it by a level and a facility, display the filtered messages to a user and save it in a log file. This module is used by all other modules of the software.

A module that describes the mathematical data structures and related functions is *Mathematical Tools*. Currently it contains realizations of the following mathematical concepts: *Vector* and *Matrix*, the general classes of linear algebra, specialized objects (like *Symmetric Matrix* or *Upper Triangular Matrix*, to optimize calculations), and classes that describe coordinate transformations in 3-Dimensional space, like *Vector3D* and *Matrix3D*.

Geodetic data structures, their relationships and relevant procedures are implemented in the module *Modeling Subsystem*. The main purpose of this module is computation of theoretical values of observables (time delay and delay rate) and their corresponded partial derivatives. The module implements the following abstractions: *Epoch* of observation and *time interval*, identities and attributes of various objects (*radio sources*, *stations* and *baselines*), VLBI *observations* and auxiliary data. It calculates ionospheric corrections, zenith delay and mapping functions, and provides models for clock breaks. This module uses data structures and algorithms from *Mathematical Tools*.

The module *Estimator* is responsible for performing Least Squares Estimation. The core of this module is a class called *Estimator* that solves systems of equations and obtains estimated parameters. Also the module provides an interface to other parts of the software to communicate with the *Estimator*, feeding it data and retrieving solutions. That is made possible with the

classes *Partial* and *Parameter*. The current realization implements estimation of unbiased, session-wide parameters. Later we will add global and stochastic parameters.

The module *I/O Subsystem* provides operations of input/output and supports various data formats. The module communicates with other modules which implement a particular format for data representation and provides a unified I/O interface to other parts of the software. Currently, only Mk3 database handling is implemented, which is realized in the module *Mk3 DBH*. This module allows one to read the content of a Mark-III DBH file into a temporary place in computer memory, get access to data and modify them, change format (add or delete a particular LCODE) and write modified data into a file in the Mark-III DBH format.

The module *Config/Setup* is a small module that describes models, system configuration and the parameter set up that should be applied in the analysis.

The module *Solution Manager* controls solution production. Depending on the configuration of the solution, it checks the available (O-C) and partials and, if necessary, calls proper procedures from the *Modeling Subsystem* to evaluate them. It asks the *Estimator* to create the solution and *Report Generator* to prepare the end user output. Finally, it provides obtained results to an end user. This module could be considered as a core of the software – it uses almost all other modules and realizes communication between different parts of the software.

The Graphical User Interface is implemented in *GUI* module. The module interacts with a user to visualize observations, solutions and auxiliary data. One part of the *GUI* module, the *Plotting Subsystem*, allows it to browse a wide spectrum of data. Also, it allows a user to modify the data. The *Plotting Subsystem* is universal and can be used in various applications. This module uses widgets from the *Qt* library.

4 Current Functionality

Currently, the software development process is in the intermediate stage and not all functions are realized. The following abilities are realized.

– *v*Solve can read VLBI observations in Mk3 DBH format, modify data and save modified information into a file on the disk. It assumes that the observations are multiband and automatically loads all bands of closest version number (if the files are available). There is no limitation on the number of bands.
– It can display various values that are stored in the files, evaluated on the fly or estimated in data analysis. Using the *Plotting Subsystem* a user can eliminate outliers, resolve ambiguities and edit clock break parameters interactively.
– The *v*Solve software has ability to compute ionospheric corrections from dual channel observations.
– The software is able to estimate the parameters of clock functions, zenith delays, stations positions and source coordinates. Also, it applies No-Net-Translate and No-Net-Rotation constraints specified by the user to stations and sources while estimating their coordinates.
– Automatic ambiguity resolution is implemented in the *v*Solve. It assumes that ambiguity spacing could vary from baseline to baseline.
– It can detect and take into account clock breaks. There are two approaches to dealing with clock breaks, manual and automatic.

5 Plans for Future

We expect that the first public release of *v*Solve will be made in the middle of 2011. To make it available, the following procedures need to be implemented in the software: 1) linear piecewise and stochastic estimation of parameters; 2) data reweighting; and 3) interaction with current CALC/SOLVE data structures.

References

S. Bolotin, J.M. Gipson and D. MacMillan. *Development of a New VLBI Data Analysis Software*. In D. Behrend, and K.D. Baver, editors, *IVS 2010 General Meeting Proceedings*, NASA CP 2010-215864, NASA GSFC, Maryland, pages 197–201. 2010.